C Programming Cheat Sheet

⇒ C is a programming language developed at AT & T'S Bell Laboratories at USA in 1972 by Dennis Ritche.

⇒ Anly programming language can be divided in two categories :-

1. Problem oriented (High level)
2. Machine oriented (low level)

(*) But C is considered as a Middle level language. (becuse it is modular, Portable & reusable)

⇒ C Programming Structure :-

```
pre-processor directives
global declarations
main()
{
        local variable declarations
        statement sequences
        function invoking

}
```

⇒ There are 32 keywords available in C Program Those keywords are known as 'reserved' keyword, whose means has already been explaned to the C compiler.

⇒ C character set :-

A character set denotes any alphabet, digit or special symbol used to represent information.

Alphabets : A, B . . . X, Y, Z, a, b, . . . x, y, z

Digits : 1, 2, 3 . . . 9, 0

Special symbol : _ ^ ' ! @ # [ ] < > , . ? / \
. . . . . so on

⇒ Rule of writing, compiling and Executing C Program.

1. C is case sensitive means variable name "Counter" is different from the variable name "counter".
2. All keywords are lowercase.
3. keywords can't be use for any purpose (Like- variable name)
4. every C Program statement must be end with ';' Thus ';' acts as a statement terminator.
5. First charactor must be an alphabet or underscore (_), no special symbol other then underscore, no commas or blank space are allowed within a variable, constant or keyword.
6. variable must be declared befour using it, in the program.
7. Program need to be compiled befour execution
8. File shoud be the extension (.c)

Variable :-

```
int x;          // A int type variable
char x = 'c';      // A char type variable , value is 'c'
float y,z = 3.56, 2.57;  // float type variable
const int x = 88; // A constant variable, can't
                      assign to after declaration
                      (compiler enforced)
```

Basic Syntex $\boxed{C}$

// Boilerplate code

```
# include <stdio.h>
int main()
  {
     return 0;
  }
```

printf() function

```
printf("Hello world!");   // Hello world
// It is used to show output on the screen
```

Scanf() function

```
Scanf ("placeholder = 'format specifier' :", address variable)
                                              operator
// It is used to take input from the user
```

Comments

```
// Single Line comment

/* Multiline comment */
```

Data Type & Placeholder

General form for declaring a variable is:

>> datatype name;

example

```
# include < stdio.h>
   void main()        // void type return null
      {
         int sum;
          sum =12;
            sum = sum + 5;
            printf("sum is %d", sum);
      }
```

// The fact %d is the placeholder for integer variable
// value that is name comes after double quotes.

placement-holder	Format
%c	char
%d	int
%f	float
%s	string of char
%p	display a pointer
%%	print a %

Escape Sequences

\n	new line
\b	backspace
\t	Horizontal tab
\"	quotation mark
\v	vartical tab
\'	Apostrophe
\\	Backslash
\?	question mark

Type Casting

(type) a ⟶ Returns a as datatype

Alarm or Beep

It produces a beep sound

\a

Conditional Statement

If Statement

syntax

```
if (/* condition */)
    {
      /* code */
    }
```

If - else statement

syntax

```
if (/* condition */)
    {
       /* code */
    }
else {
       /* code */
    }
```

If else-if statement

Syntax

```
if ( /* condition */)
    {
         // statement
    }
else if ( /* condition */ ){
         // statement
    }
else {
         //statement
    }
```

Switch

```
Switch( expression)
    {
     case   constant-expression:
       Statement1;
       Statement2;
       break;

     case   constant-expression:
         Statement1;
          break;
       ......
       default :
             Statement-1;
              break;
          }
```

Nested if-else :-

```
            if ( expression1)
                      Statement;

            else
              {
                     if ( expression2)
                              Statement;

                     else
                        {
Outer     Inner
block     block                 block af statement;

                         }

                 }
```

There are some things that you simply can't do with a switch statement :-

1. A float expression cannot be tested using switch
2. Case can never have variable expression (ex: x+3)
3. Multiple cases can't use same expression.

Iterative Statement

While loop

```
while (/* condition */)
  {
    /* code */
  }
```

Do-While loop

```
do {
    /* code */
  } while (/* condition */)
```

for loop

```
int count = 5;
for( int i = 0; i < count; i++)
     {
          /* code */
     }
```

// First starting point declaration
// second condition check
// Third incrementation / declaration.

Break Statement-

break keyword inside the loop is used to terminate the loop.

Continue Statement-

Continue statement's mean continue keyword skips the rest of the current iteration of the loop and returns to the starting point of the loop.

```
[ Continue ; // It does not terminate the
             // Loop however.
```

loop details

For loop

```
For( initialization; test condition; run every time command).

    {
        /* code */

    }
```

while loop

```
/* initialization */
while ( expression)
      {
        /*statement */
      } /* increment /decrement */
```

do-while loop

```
do
  {
     Statement ;
  } while ( expression);
```

Note:- ensure that the statement within the loop execute at last onece.

Goto Statement

Goto syntax

```
for ( .... )
{
    for ( .... )
    {
        ....
            if ( disaster)
            {
                goto error;
            }
    }
}
....
error :
    // clean up the mess
```

Function & Recursion

Function is combined of a block of code that can be called or used anywhere in the program by calling the name. Body of a function starts with { and ends with }. This is similar to the main function ...

basic syntax of functions

```
# include <stdio.h>
   myfunc(); // function prototypes
main()
 {
     my func();

 }
myfunc() {    // function defination
      printf(" Hello, this is a test-");
   }
```

Note :-

1. Any c program contains at last one function
2. If a program has only one function, it must be main() function.
3. Program execution always begains from main() function.
4. After each function has done its things, control returns to main(). when main() run out of function calls, the program ends.

Function Argument-

Functions are able to accept input parameters in the form of variables. These input parameter variables can then be used in function body.

Syntax of function Argument-

```
#include <stdio.h>
/* use function prototypes */
Sayhello ( int count);
main()
  {
        Sayhello (4);
  }
 Sayhello (int count)
     {
        int c;
        for (c = 0 ; c < count ; c++)
                printf(" Hello \n");
     }
```

Based on the examples we have called Sayhello() function with the parameter '4'. This function recives an input value and assigs it to 'count' variable befour starting execution of function body. 'Sayhello()' function will then print a hello massage count times on the screen.

Function return value.

Syntex

```
int sum ()
  {
      int a, b, c ;
          a = 1;
          b = 4;
          c = a+b;
          return c
  }
```

vaid return type

here return null

```
vaid test()
  {
      /* code */

  }
```

Recursion

In C program for the functions to call themself. A function call recursive if a Statement within the body of a function calls the same function.

» Best example of recursive function →

Factorial

$5! = 5 \times 4!$

$= 5 \times (4 \times 3!)$

$= 5 \times 4 \times (3 \times 2!)$

$= 5 \times 4 \times (3 \times (2 \times 1!))$

$= 5 \times 4 \times 3 \times 2 \times (1 \times 0!)$

$= 5 \times 4 \times 3 \times 2 \times 1 \times 1$ [∵ 0! = 1]

$= 5 \times 4 \times 3 \times 2 \times 1$

$= 120$

```
# include <stdio.h>
    int rec ( int ); //Prototype declaration
void main()
        {
            int a, fact;
            printf ("\n Enter only any number :");
            Scanf (" %d", & a);
            fact = rec (a); // function call
            printf(" Factorial value = %d", fact);
        }
        int rec ( int x) // function defination
        {
            int f;
            if ( x == 1)
                    return (1);
            else
                    f = x * rec (x-1);
                    return (f);
        }
```

Explanation

if a = 5

rec(5) , returns (5 * rec(4),
which returns (4 * rec(3),
which returns (3 * rec(2),
which returns (2* rec(1),
which returns (1)))))

call by value

```
# include <stdio.h>
void test (int a);
void main()
   {
       int m;
       m = 2;
       printf("\n M is %d", m);
       test(m);
       printf("\n M is %d \n", m);
       return 0;

   }
void test (int a)
   {
       a = 5;
   }
```

```
M is 2
M is 2
```

// value not change.